Analysis of Step-Reduced SHA-256*

Florian Mendel**, Norbert Pramstaller,
Christian Rechberger and Vincent Rijmen

Christian.Rechberger@iaik.tugraz.at

Institute for Applied Information Processing and Communications (IAIK)
Graz University of Technology, Austria

www.iaik.tugraz.at/research/krypto

Abstract. This is the first article analyzing the security of SHA-256 against fast collision search which considers the recent attacks by Wang *et al.* We show the limits of applying techniques known so far to SHA-256. Next we introduce a new type of perturbation vector which circumvents the identified limits. This new technique is then applied to the unmodified SHA-256. Exploiting the combination of Boolean functions and modular addition together with the newly developed technique allows us to derive collision-producing characteristics for step-reduced SHA-256, which was not possible before. Although our results do not threaten the security of SHA-256, we show that the low probability of a single local collision may give rise to a false sense of security.

1 Introduction

After recent cryptanalytic results on MD5 [19], SHA-1 [2, 14, 18] and similar hash functions, the resistance of members of the SHA-2 family (*i.e.* SHA-224, SHA-256, SHA-384 and SHA-512) [12] against recent attacks is an important issue.

While SHA-1 and MD5 are currently the most commonly used hash functions worldwide, the direct successor of SHA-1, SHA-256 is in many cases considered to be an upgrade option. However, SHA-256 did not receive as much cryptanalytic scrutiny from the cryptographic community as other hash functions. Even though the underlying design principle did not change since MD4, SHA-256 needs to be considered separately. It is expected to be much stronger than SHA-1, but several questions concerning its collision resistance need to be answered:

- Are the currently known techniques applicable to SHA-256? Which ones and to what extent?
- What about new techniques which are specifically designed to be applied to SHA-256?

In this article, we give preliminary answers to these questions. To put our contribution into perspective, we first survey existing approaches and previous results.

* The work in this paper has been supported by CRYPTREC.
** This author is supported by the Austrian Science Fund (FWF) project P18138.

1.1 Outline of existing approaches

The basic approach for efficient collision search of the predecessors of SHA-256, SHA-0 and SHA-1, can be described as follows:

1. Identify local collisions in the state-update transformation.
2. Search for low-weight perturbation vectors by searching for low-weight expanded messages. In the approach by Chabaud and Joux [4] the perturbation vectors need to satisfy some additional properties which were dropped later on by Wang *et al.* [18] by using more complicated techniques.
3. Build the difference vector by interleaving the local collisions as described by the perturbation vector. Note that in [14] and [13] an approach is described which combines the three steps above.
4. The complexity of the collision search attack is related to the probability with which the characteristic described by these interleaved local collisions is followed.
5. By adjusting message bits for the chosen characteristic and allowing small variations in the characteristic, the computational effort for the collision search is decreased.

1.2 Survey of existing results on SHA-256

Being standardized by NIST in 2000 [12], the first published independent analysis of members of the SHA-2 family was done by Gilbert and Handschuh [6]. They show that there exists a 9-step local collision with probability 2^{-66}. Later on, the result was improved by Hawkes *et al.* [7]. By considering modular differences, they give a new maximal probability of 2^{-39}.

In [9] SHA-256 is analyzed in encryption mode. Attacks based on related-key assumptions for up to 37 steps are presented there. In [21], all modular additions are replaced by XOR. For this variant, a search for pseudo-collisions which is faster than brute force search for up to 34 steps faster is described.

In [11] a variant of SHA-256 is analyzed where all Σ and σ are removed. The conclusion is that collisions can be found much faster than by brute force search for this variant. Additionally, some low-weight expanded message differences for a GF(2)-linearized message expansion are given. The work shows that the approach used by Chabaud and Joux [4] in their analysis of SHA-0 is extensible to that particular variant of SHA-256.

1.3 Our contribution

So far, nobody described ways to do collision search attacks for the unmodified SHA-256 or step-reduced variants thereof. After a short description of SHA-256 in Sect. 2, we address this issue in several ways.

First we analyze the message expansion of SHA-256 and show that its properties prevent the efficient extension of the techniques used by Chabaud and Joux [4] on SHA-0 or by Wang *et al.* in the analysis of SHA-1 [18] (see Sect. 3.1).

Table 1. Notation

notation	description
$A_i \dots H_i$	state variables at step i of the compression function
$A \oplus B$	bit-wise XOR of state variable A and B
$A + B$	addition of state variable A and B modulo 2^{32}
A'	XOR difference in state variable A
M_t	input message word t (32 bits), $t \geq 1$
W_t	expanded input message word t (32 bits), $t \geq 1$
$ROTR^n(A)$	bit-rotation of A by n positions to the right
$SHR^n(A)$	bit-shift of A by n positions to the right
N	number of steps of the compression function

To illustrate our point, we define a variant SHA-256-3R, where the shift operation in the message expansion is replaced by a rotation. We show how to use low-weight perturbation vectors in a collision search for this variant in Sect. 3.2.

Next we focus on the unmodified SHA-256. To overcome the limitations we identified before, we introduce the idea to drop the requirement of having a perturbation vector which is a valid expanded message. In Sect. 3.3, we develop a way to derive this new type of perturbation vector for SHA-256, which can be used to find collision-producing characteristics with high probability. The price we have to pay is an increase of the search space for this new type of perturbation vector.

Some heuristics, which were developed in the case of SHA-1 to reduce the search space, do not apply to SHA-256. In Sect. 3.4 we describe a new way to reduce the search space.

As an example, a 19-step collision for unmodified SHA-224 is presented in Sect. 3.5, including a detailed description of the characteristic being used. There are two lessons to be learned from this example. Firstly, compared to independently multiplying probabilities for local collisions, interleaving them dramatically increases their overall probability. This contrasts with observations on older members of the SHA family. Secondly, techniques that exploit the combination of Boolean functions and modular addition in the state update are shown to be applicable in this example of a SHA-224 collision.

Additionally, we briefly survey methods to speed up the collision search for SHA-256 in Sect. 4. These methods and refinements thereof are subsequently used in some examples. These examples include the fast generation of 18-step collisions and 22-step pseudo-collisions with non-zero message difference on a standard PC.

2 Description of SHA-256

In the remainder of this article we use the notation given in Table 1. A complete description of SHA-256 can be found in [12]. We briefly review parts of the specification needed subsequently.

SHA-256 is an iterated cryptographic hash function based on a compression function that updates the eight 32-bit state variables $A, \ldots, H$ according to the values of 16 32-bit words $M_0, \ldots, M_{15}$ of the message. The compression function consists of 64 identical steps as presented in Fig. 1. The step transformation employs the bitwise Boolean functions f_{MAJ} and f_{IF}, and two GF(2)-linear functions

$$\Sigma_0(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)\,,$$
$$\Sigma_1(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)\,.$$

The i-th step uses a fixed constant K_i and the i-th word W_i of the expanded message. The message expansion works as follows. An input message is padded and split into 512-bit message blocks. Let ME denote the message expansion function. ME takes as input a vector M with 16 coordinates and outputs a vector W with N coordinates. The coordinates W_i of the expanded vector are generated from the initial message M according to the following formula:

$$W_i = \begin{cases} M_i & \text{for } 0 \le i < 16 \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & \text{for } 16 \le i < N \end{cases} . \qquad (1)$$

Taking a value for N different to 64 results in a step-reduced (or extended) variant of the hash function. The functions $\sigma_0(x)$ and $\sigma_1(x)$ are defined as follows: $\sigma_0(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$ and $\sigma_1(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$.

3 Finding collision producing characteristics for step-reduced SHA-256

Finding collision-producing characteristics for SHA-256 with a high probability is difficult. While searching for high probability characteristics, $GF(2)$-linear approximations that hold with high probability are useful. In the case of the addition mod 2^{32}, bit-wise XOR of the inputs has probability 1 for the LSB and probability 0.5 for all other bits. In the case of the Boolean functions f_{IF} and f_{MAJ}, several approximations are possible which all hold with probability 0.5. Superficially comparing the results in [7] and [6][1] would lead to the preliminary conclusion that the notion of modular differences instead of XOR differences offers a significant advantage. However, we argue that this is not the case. Using the XOR differences as presented in [6] and looking at possible characteristics

[1] Probability of 2^{-39} vs. 2^{-66} for a single local collision in SHA-256

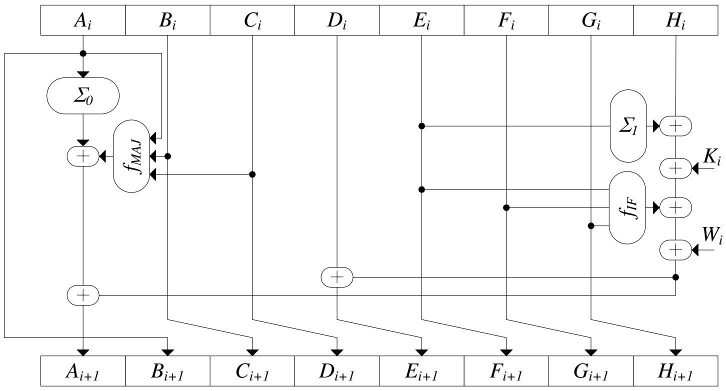

Fig. 1. One step of the state update transformation of SHA-256.

for a local collision, we estimate that the probability of a single local collision (depending on the bit position of the perturbation) can be higher than 2^{-39}. This is by assuming unknown state variables at the beginning of the local collision. Since this compares favorably with the best results known so far, we will stick to XOR differences..

Compared to SHA-1, where the corresponding probabilities are between 2^{-2} and 2^{-5}, the probability for a local collision in SHA-256 is still very low. However, we will show by means of an example, that by interleaving several local collisions to build a collision-producing characteristic, the combined probability is much higher than the product of the single probabilities. This effect also occurs in the case of SHA-1 [18], but with much less impact on the overall complexity of the attack. However, before we arrive there, we need to discuss how to find suitable ways for interleaving these local collisions.

3.1 Why existing approaches do not work

In this section we discuss to which extent the methods that are used in the analysis of SHA-0 and SHA-1 are applicable to SHA-256 as well. A very high-level description of the so-called perturbation-correction method to find collisions for SHA-0 and SHA-1 used in [4, 18] could be the following:

1. Find a vector d such that the perturbation vector $d' = \mathrm{ME}(d)$ has a low Hamming weight.
2. Determine the correction vectors c'_u which ensure that the expanded message difference $e' = d' + \sum_u c'_u$ results in a collision for the linearized hash function. The mapping from d' to the c'_u depends on the properties of the state update transformation alone.
3. Determine the vectors c_u such that $c'_u = \mathrm{ME}(c_u)$. Construct the message difference as $e = d + \sum_u c_u$.

4. Determine M and M^* such that the differences in the real hash function follow the characteristic built for the linearized hash functions. We will refer to this characteristic as *L-characteristic*.

For all the hash functions of the SHA family, the vectors c'_u can be computed as $c'_u = \mathrm{R}_{r_u} \circ \mathrm{T}_u(d')$. The map $\mathrm{R}_{r_u}(x')$ *rotates* every coordinate of the vector x' over the constant amount r_u. The map $\mathrm{T}_u(x')$ *translates* the coordinates of the vector x' over u positions to the right, dropping the leftmost u coordinates and filling in u zeroes on the left. The values (u, r_u) depend on the state update transformation. For instance, for the case of SHA-1, the values are $(1,5),(2,0),(3,-2),(4,-2)$ and $(5,-2)$.

The message expansion of a hash function is not surjective. We call x' a *valid expanded message* if there exists a value x such that $x' = \mathrm{ME}(x)$. Additional conditions can be imposed on d in order to ensure that the vectors c'_u are valid expanded messages. In particular, we need the two following conditions.

Condition 1: $\mathrm{R}_{r_u}(\mathrm{ME}(d))$ needs to be a valid expanded message, for all values r_u that occur.

Condition 2: $\mathrm{T}_u(\mathrm{ME}(d))$ needs to be a valid expanded message, for all values u that occur.

It can easily be verified that for the message expansion of SHA-1, Condition 1 is satisfied for all d and for all r_u. Condition 2 can be satisfied by ensuring that "the backwards expanded difference equals zero in the first 5 steps" [4].

For the case of SHA-256 with linearized message expansion (all modular additions are replaced by XOR), Condition 2 can easily be satisfied by requiring that the backwards expanded difference equals zero in the first 8 steps. Contrary to SHA-1, satisfying Condition 1 imposes severe restrictions on d.

It has been observed before [13, 14] that the perturbation-correction method imposes overly strict requirements. Indeed, instead of requiring that d' and each of the c'_u are valid expanded messages, it suffices to demand that the sum $e' = d' + \sum_u c'_u$ is a valid expanded message. For SHA-1, this observation doesn't lead to improved results. However, for SHA-256, it does as we will show in Sect. 3.3.

We show that for SHA-256, Condition 1 cannot always be met by proving the following Theorem.

Theorem 1. *For SHA-256, not all perturbation vectors d satisfying Condition 1 lead to a perturbation-correction vector e' which is a valid expanded message.*

The proof is given in Appendix A and shows first that this holds for a variant of SHA-256 with linearized message expansion and then extends this result to unmodified SHA-256.

The implication of this result is as follows: when we try to extend the standard perturbation-correction method, which is at the core of every analysis of SHA-0 and SHA-1 including those of Wang *et al.*, to analyze SHA-256, we cannot prevent the fact that there will be *unwanted differences* due to the message expansion. For later reference, we term them "ghost differences of type 2".

Theorem 1 also shows that the additional degrees of freedom we have due to the $GF(2)$ non-linearity of the message expansion are not sufficient to always correct this undesired behavior. In other words, by applying the standard perturbation-correction method, we are facing impossible differentials in the message expansion.

Implications of Theorem 1 on the collision search complexity. The major improvement of Wang *et al.*, which eventually lead to the break of SHA-1, was the ability to deal with a different kind of ghost-difference. By dropping Condition 2, unwanted differences appear in the first 5 steps of SHA-1. We term them "ghost differences of type 1". In the following, we expand on that.

In the case of SHA-1 [18], the (near-)collision-producing characteristic is actually a concatenation of a low-probability general characteristic with a high probability L-characteristic. By means of the general characteristic in the first steps, these "ghost differences of type 1" are incorporated. This general characteristic has a very low probability, but this fact is compensated by message modification, which "bypasses" the probability of the chosen characteristic for more than 20 steps.

What would happen if we drop Condition 2 in the case of SHA-256? The "ghost differences of type 1" as described above will now appear up to step 8. However, starting from step 17 until step N, there will also be "ghost differences of type 2". Even if it would be possible to incorporate them in an even more complex general characteristic covering all N steps, the impact of this approach on the attack complexity would be severe.

The attack complexity is determined by the probability with which the part of the characteristic is followed that is not covered by message modification techniques. Since the low-probability general characteristic needs to be followed for all steps now, message modification cannot prevent its influence on the attack complexity anymore. It is by no means clear that such a general characteristic for all N steps of SHA-256 is even possible. Even if it is, the probability to follow this general characteristic up to step N is likely to be prohibitively low. Therefore, an other approach will be needed.

3.2 A short detour: SHA-256-3R

We show that by making a small change in SHA-256, the basic perturbation-correction approach can be applied again. We name this variant SHA-256-3R and change the message expansion of SHA-256 in the following way: The functions $\sigma_0(x)$ and $\sigma_1(x)$ are replaced by the following:

$$\sigma_0(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus ROTR^3(x)$$
$$\sigma_1(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus ROTR^{10}(x)$$

SHR is replaced by $ROTR$ which has the effect that Condition 1 imposes no restrictions anymore. Table 2 gives us the starting point for our analysis. It shows a remarkably low-weight perturbation vector which satisfies the following requirements. Firstly, the last 8 perturbation words are all-zero, which means that we can finish all the needed corrections. Secondly, the backwards expansion is all-zero for the first 8 steps which prevents "ghost differences of type 1" in our perturbation-correction vector. These requirements are enough to build a

Table 2. Low-weight expanded message for the XOR-linearized 31-step message expansion of SHA-256 which can be used as a perturbation vector for SHA-256-3R

d'_i			
$i = 1$	80000000	$i = 11$	0
$i = 2$	11002000	$i = 12$	0
$i = 3$	80000000	$i = 13$	0
$i = 4$	14044aa8	$i = 14$	0
$i = 5$	00205000	$i = 15$	0
$i = 6$	0	$i = 16$	0
$i = 7$	0	$i = 17$	0a020000
$i = 9$	0	$i = 18$	0
$i = 9$	11002000	$i = 19$	80000000
$i = 10$	80000000	$i = 20 \ldots 31$	0

collision-producing characteristic which is constructed by interleaved local collisions as described by the perturbation vector. It is given in Appendix B. Note that this characteristic is an L-characteristic.

Most of the local collisions will be completed within the first 16 steps. The last local collision will be completed at step 27. Due to the small change in the message expansion, we do not have any "ghost differences of type 2". Showing this fulfills the purpose of this detour, hence we stop the analysis of this L-characteristic of SHA-256-3R here.

3.3 Extending the Rijmen-Oswald approach to unmodified SHA-256

In the next two subsections, we show that despite the findings in the previous Sect. it is still possible to find perturbation vectors of low-weight which lead to collision-producing characteristics. It will turn out that these perturbation vectors are no longer valid expanded messages by themselves. Thus we will have a *new type of perturbation vector* for SHA-256.

The approach to find such a new type of perturbation vector for SHA-256 is outlined below. The underlying idea is originally proposed in [14] and extended in [13]. Basically, it works as follows. First, we build a linearized version of the message expansion and the state update transformation. Then, we construct a generator matrix G which describes all possible state variables that result in a collision for this linearized version. By searching for low-weight codewords(see [3, 10, 15]) in the linear code described by G, we are actually searching for L-characteristics with high probability.

In the case of SHA-256, the dimension of G is $512 \times (9 * 32 * N)$ where N denotes the number of steps. Note that we can cut the parts representing the state variables B, C, D, F, G, H and thus reducing the *length* of the code without loosing information. However, without a way to reduce the *size* of the code and not excluding low-weight codewords, a search for L-characteristics with high probability is not feasible.

3.4 Reducing the search space to find useful L-characteristics

In the analysis of SHA-1 [2, 8, 14, 18], it was possible to reduce the search space for perturbation vectors or general collision producing characteristics by applying the following observation. Low-weight expanded messages for SHA-1 have the property that non-zero bits occur in bands, *i.e.* the non-zero values are concentrated on a few bit positions in every word. This can be explained by the weak avalanche effect of the SHA-1 message expansion. This heuristic does not apply to SHA-256. The functions $\Sigma_0, \Sigma_1, \sigma_0$ and σ_1 effectively prevent such a structure in L-characteristics. Therefore, the search space and hence the size of the code needs to be reduced by other means.

Another way of looking at the search for low-weight codewords in the code describing L-characteristics is as follows. Searching for low-weight codewords maps to searching for low-weight solutions in a homogeneous system of equations in GF(2). Actually, the corresponding check matrix H of the code described by G is a representation of the coefficients of this system. The variables refer to all message bits and state variable bits in the linearized variant. The system of equations described by H is under-defined, *i.e.* there are more variables than equations.

Forcing bits to zero or one can also be seen as adding new equations, where we simply set this bit to that particular value. The generator matrix G as described in 3.3 gives us 512 degrees of freedom, which means we can add up to 511 equations to H. By forcing those bits to zero which we expect to be zero in an L-characteristic, we eventually arrive at a system of equations where it is feasible to search for low-weight solutions. Note that this is a rather rough way to reduce the search space which does not work for larger number of steps N.

3.5 Example of a collision-producing L-characteristic for 19-step SHA-224

In Table 3, we give an L-characteristic of a 1-near-collision for 19 steps of SHA-256 which is a 19-step collision for SHA-224 at the same time. Note that the only difference between SHA-224 and SHA-256 is that at the output, the rightmost 32 bits are discarded. By applying the techniques described in Sect.3.4, we reduced the size of the code to 64, which led to our results.

The perturbation vector which is used as a building block for this characteristic is the vector A' in Table 3. The perturbation vector is not a valid expanded message. Note that this perturbation vector can be word-wise rotated without loosing its property of leading to a perturbation-correction vector which is always a valid difference between expanded message. Thus, we can rotate our L-characteristic to maximize the number of MSBs involved such that the probability of this L-characteristic is maximized. The first perturbations start at step

Table 3. Example of a 19-step SHA-224 collision. All-zero differences are denoted by a single 0 to improve readability

Step	W'	A'	B'	C'	D'	E'	F'	G'	H'
1-4	0	0	0	0	0	0	0	0	0
05	85009008	85009008	0	0	0	85009008	0	0	0
06	a14cae12	a1442610	85009008	0	0	02000802	85009008	0	0
07	0	0	a1442610	85009008	0	084c4120	02000802	85009008	0
08	8200a8a8	00000020	0	a1442610	85009008	00000020	084c4120	02000802	85009008
09	85009008	85009008	00000020	0	a1442610	01008008	00000020	084c4120	02000802
10	0	0	85009008	00000020	0	02000802	01008008	00000020	084c4120
11	0	0	0	85009008	00000020	0	02000802	01008008	00000020
12	0	00000020	0	0	85009008	0	0	02000802	01008008
13	0	0	00000020	0	0	84001000	0	0	02000802
14	00088802	0	0	00000020	0	0	84001000	0	0
15	0	0	0	0	00000020	0	0	84001000	0
16	0	0	0	0	0	00000020	0	0	84001000
17	0	0	0	0	0	0	00000020	0	0
18	0	0	0	0	0	0	0	00000020	0
19	0	0	0	0	0	0	0	0	00000020

5, and there are 23 of them in total. The local collision for SHA-256 as originally described in [6] needs 24 single-bit differences in the message. Using this as an upper bound, we would expect up to 552 single-bit differences in the expanded messages. Note that the actual value 37, the weight of the message difference, is far below this upper bound.

In this particular example, more than 200 conditions on the state variables need to be met to follow the given L-characteristic. Assuming random independent trials, each perturbation would on average contribute a factor of 2^{-10} instead of about 2^{-40} to the overall probability. Hence, the fact that a single local collision in SHA-256 has a comparatively low probability may give a false feeling of security. Note that the actual collision search complexity can be further reduced by techniques mentioned in Sect. 4.

3.6 Adjustments to circumvent impossible characteristics

In this section we take a closer look at the presented L-characteristic. The best probabilities for local collisions are achieved by approximating the differential behavior of the functions f_{MAJ} and f_{IF} by 0. On average, both approximations hold with probability 0.5. However, in certain cases, the probability for this approximation is 0.

Translating these properties into the sequence of states of the SHA-256 compression function gives rise to the following observations.

Observation 1 *Whenever we have 3 non-zero differences in consecutive variables of the state (A'_r, A'_{r+1}, A'_{r+2}) at the same bit position, the chosen linear approximation fails to predict any subsequent difference.*

Observation 2 *Whenever we have 2 non-zero differences followed by one zero differences in consecutive variables of the state (E'_r, E'_{r+1}, E'_{r+2}) at the same bit* *position, the chosen linear approximation fails to predict any subsequent difference.*

In order to prevent these cases, we would need to exclude all of them from our search space. However by doing this, low-weight solutions might be excluded. By using the degrees of freedom we have in our characteristic, *i.e.* various ways in which differences can propagate through the Boolean functions and the modular addition, we observe the following. It turns out to be possible to circumvent these impossible characteristics by choosing a slightly different characteristic for the same differential. Note that a similar strategy was used in the analysis of SHA-1 [2, 18].

This suggests that the additional complexity of the SHA-256 state update transformation does not prevent us from using a similar approach. To illustrate this property, we take the 19-step L-characteristic presented in the previous subsection. Indeed, we have a single case of two consecutive words which have a difference at the same bit position. This happens in E_7 and E_8 at bit position 5. Thus Observation 2 applies. The result is that the function f_{IF} accepts $(0, 1, 1)$ as input difference at bit position 5 in step 9. Hence the output of f_{IF} will flip with probability 1.

The easiest way to cancel out this additional difference is by using other differences in the same step. At the output of Σ_1, we have a difference in bit 4. By a simple carry extension we can produce a change in the carry caused by this difference. The result will be that in contrast to the prediction of our L-characteristic, the difference in bit 4 will cause bit 5 to flip as well. However, this additional difference due to the carry extension will now cancel the additional difference at the output of f_{IF} in this step. Eventually, the path described by the L-characteristic can be followed without the need to circumvent additional impossible characteristics.

4 Increasing the performance of collision search for SHA-256

In this section we briefly cover ways to speed up the collision search for members of the SHA-2 family once a suitable characteristic is found. For their predecessors SHA-1 and MD5, two competing approaches can be found in the literature. One approach has been termed message modification. It was first introduced in [16, 19]. A variant of the technique was also used in the most recent analysis of SHA-0 [20] and SHA-1 [17, 18].

The second approach was introduced in [1] and later on applied in [2]. It extends the idea of [5] to the hash function SHA-0. So-called neutral bits in the input message are used to circumvent the probabilistic behavior of the first steps of SHA-0. Within certain limits, both approaches can be extended to the case of SHA-256. Subsequently, we briefly discuss to which extent this is possible.

In the first 16 steps of SHA-256, the conditions on the state variables can be directly rewritten to conditions on the message words. The procedure can be described as follows:

$$
\begin{aligned}
A_{N+1} &= f_1(A_N, \ldots, H_N) + K_N + W_N \\
E_{N+1} &= f_2(A_N, \ldots, H_N) + K_N + W_N
\end{aligned}
\tag{2}
$$

Next, adjust A_{N+1} and E_{N+1} accordingly to meet the conditions derived for the characteristic. Then calculate

$$
\begin{aligned}
W_N &= A_{N+1} - f_1(A_N, \ldots, H_N) - K_N \\
W_N &= E_{N+1} - f_2(A_N, \ldots, H_N) - K_N
\end{aligned}
\tag{3}
$$

Note that by applying these formulas, each new message word is calculated twice. Hence it is possible that changes in the message bits contradict each other. In these cases, adjusting message words which are input in the steps before the contradiction occurs is necessary. A high-level algorithm to deal with this issue is given below.

Algorithm 1 Way to fulfill contradicting conditions in SHA-256

Require: Contradicting Conditions in A_i and E_i
Ensure: Condition in A_i and E_i are fulfilled
  Fulfill Condition in E_i by adjusting W_i as described in Equation (3).
  while Condition in A_i is not fulfilled or any previously fulfilled conditions are affected **do**
    Go back to step $x \in \{0 \ldots i-1\}$ and check if W_x can be adjusted such that the condition in A_i is fulfilled
  end while

Note that such methods were not needed in any of the predecessors of the SHA-2 family, because there can be no contradictions in fulfilling conditions in the first 16 steps.

In order to illustrate the technique, let's assume that by applying the simple message modification rules described by (2) we are getting a contradiction in bit 4 of W_6. We set this bit such that the condition on state variable E_6 is met (Step 1). In order to fulfill the condition on state variable A_6 (*i.e.* bit 4 should have opposite value) we simply go back one step and flip bit 4 in W_5 (Step 3). That way, bit 4 in A_5 is flipped. However A_5 is not directly influencing A_6, but via Σ_0 and the f_{MAJ}-function. The effect is twofold.

- Firstly, depending on the other inputs of the f_{MAJ}-function, the output might not change. In this case, B_5 or C_5 need to be updated by going back and adjusting the input word at the respective step. In general, every message word before the step where the contradiction occurred might be a candidate for message modification. However, the risk that other conditions are affected by these adjustments increases with the number of backward-steps.
- Secondly, due to Σ_0, three other bit positions are also affected with every message word adjustment. These might in turn affect other conditions and

Table 4. Summary of examples

function	steps	type	local collisions	probability
SHA-256-3R	31	collision	25	-
SHA-256	18	collision	1	$\sim$ 1 using neutral bits
SHA-224	19	collision	23	$< 2^{-200}$ before message modification
SHA-256	19	1-near-collision	23	$< 2^{-200}$ before message modification
SHA-256	22	pseudo-collision	-	$\sim$ 1 using neutral bits

might even cancel out the desired effect of the flipped bit at the input via carry propagation.

If it turns out that it cannot be prevented that other conditions are affected with these adjustments, another choice in step 2 needs to be made.

To sum up, compared to SHA-1 or MD4/MD5, message modification is more complex due to the fact that two state variables are updated at the same time. After step 16, chances that existing conditions are affected by message modification increase. In the Appendices C and D we show simple examples of the application of these techniques. Table 4 summarizes them.

5 Conclusions and future work

When the attack techniques that were used successfully against SHA-1, are applied to SHA-256, several problems arise. Firstly, the shift operations in the message expansion of SHA-256 severely limit the usefulness of the perturbation-correction approach. To circumvent this obstacle, we introduced a new type of perturbation vector. We showed that it is still possible to find low-weight difference vectors that may result in a collision, but the search space increases dramatically. In order to find collisions for versions with more than 20 steps, we need new heuristics to reduce the search space.

Secondly, the increased Hamming weight of the difference and the presence of two nonlinear Boolean functions in each step make it very difficult to avoid consecutive ones in the inputs of the Boolean functions. Hence we have to deal with the fact that the linear approximations for these functions often won't hold. We have presented some ideas on how to deal with this problem.

Thirdly, the very low probability for one local collision in SHA-256 may give rise to a false feeling of security. We have shown with examples that the interleaving of local collisions results in many canceled differences. Hence the probability of n interleaved local collisions is typically significantly larger than the probability of one local collision, raised to the power n. We need to develop better ways of estimating this probability.

Acknowledgements

We would like to thank Christophe De Cannière and Krystian Matusiewicz for helpful comments.

References

1. Eli Biham and Rafi Chen. Near-Collisions of SHA-0. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
2. Eli Biham, Rafi Chen, Antoine Joux, Patrick Carribault, Christophe Lemuet, and William Jalby. Collisions of SHA-0 and Reduced SHA-1. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 36–57. Springer, 2005.
3. Anne Canteaut and Florent Chabaud. A New Algorithm for Finding Minimum-Weight Words in a Linear Code: Application to McEliece's Cryptosystem and to Narrow-Sense BCH Codes of Length 511. IEEE Transactions on Information Theory, 44(1):367–378, 1998.
4. Florent Chabaud and Antoine Joux. Differential Collisions in SHA-0. In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462, pages 56–71. Springer, 1998.
5. David Chaum and Jan-Hendrik Evertse. Crytanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *LNCS*, pages 1–16. Springer, 1986.
6. Henri Gilbert and Helena Handschuh. Security analysis of SHA-256 and sisters. In Mitsuru Matsui and Robert Zuccherato, editors, *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003, Revised Papers*, volume 3006 of *Lecture Notes in Computer Science*, pages 175–193. Springer, 2003.
7. Philip Hawkes, Michael Paddon, and Gregory G. Rose. On corrective patterns for the SHA-2 family. Cryptology ePrint Archive, Report 2004/207, August 2004. `http://eprint.iacr.org/`.
8. Charanjit S. Jutla and Anindya C. Patthak. A Matching Lower Bound on the Minimum Weight of SHA-1 Expansion Code. Cryptology ePrint Archive, Report 2005/266, 2005. `http://eprint.iacr.org/`.
9. Jongsung Kim, Guil Kim, Sangjin Lee, Jongin Lim, and Jung Hwan Song. Related-Key Attacks on Reduced Rounds of SHACAL-2. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *LNCS*, pages 175–190. Springer, 2004.
10. Jeffrey S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.
11. Krystian Matusiewicz, Josef Pieprzyk, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of simplified variants of SHA-256. In *Proceedings of WEWoRC 2005*, LNI P-74, pages 123–134, 2005.
12. National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002. Available online at `http://www.itl.nist.gov/fipspubs/`.

13. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In Nigel P. Smart, editor, *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, volume 3796 of *LNCS*, pages 78–95. Springer, 2005.
14. Vincent Rijmen and Elisabeth Oswald. Update on SHA-1. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *LNCS*, pages 58–71. Springer, 2005.
15. Jacques Stern. A method for finding codewords of small weight. In G. Cohen and J. Wolfmann, editors, *Coding Theory and Applications, 3rd International Colloquium, Toulon, France, November, 1988, Proceedings*, volume 388 of *LNCS*, pages 106–113. Springer, 1989.
16. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.
17. Xiaoyun Wang, Andrew Yao, and Frances Yao. New Collision Search for SHA-1, August 2005. Presented at rump session of CRYPTO 2005.
18. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
19. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
20. Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision Search Attacks on SHA-0. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 1–16. Springer, 2005.
21. Hirotaka Yoshida and Alex Biryukov. Analysis of a SHA-256 variant. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography (SAC 2005), Kingston, Ontario, Canada, August 11-12, 2005, Proceedings to appear*, LNCS. Springer, 2005.

A Proof of Theorem 1

Before proofing Theorem 1, we need the following Observation.

Observation 3 *Let (V',X',Y',Z') be the XOR-difference of 4 inputs of an addition mod* 2^n*. Let R' denote the XOR-difference of the result of this addition.* $\forall j$*:* $32 > j > i$ *where the following situation occurs:*

$$V'_i = X'_i = Y'_i = Z'_i = R'_i$$

the following relation must hold:

$$V'_{i+1} \oplus X'_{i+1} \oplus Y'_{i+1} \oplus Z'_{i+1} \oplus R'_{i+1} = R'_i$$

Now we can give a constructive proof for Theorem 1.

Proof. Let's first consider the case of a GF(2)-linearized variant of ME, ME_{lin}. In order to proof Theorem 1 for this variant, it suffices to show that for a particular d, the vector e' is not a valid expanded message.

We choose d to be the perturbation vector shown in Table 2. Note that A' contains the perturbation vector in this table. Vector 17 of e' (the sum of the perturbations and corrections for W_{17}) is `1b022000`. However, applying the recurrence relation for the SHA-256 message expansion, W_{17} turns out to be `8b022000`. For the GF(2)-linearized variant of the SHA-256 message expansion in Equation 1, the proof would already be finished. Let's now consider unmodified SHA-256.

We can build up on the previously proved part on the linearized variant, but need to consider the additional degrees of freedom we have due to carries. Here we need to show that no two expansions of messages m and m^* can exist such that $W_{17} \oplus W_{17}^* =$ `1b022000`.

W_{17}' can be rewritten as $(V+X+Y+Z) \oplus (V^*+X^*+Y^*+Z^*)$ Considering the recurrence relation given in Equation 1 and inserting the value from Table 2 we get $V' = \sigma_1(W_{15}) =$ `81609048`, $X' = W_{10} =$ `04f61081`, $Y' = \sigma_0(W_2) =$ `8e94a0c9`, $Z' = W_1 =$ `80000000`.

Now we apply Observation 3 and see that we get the following contradiction at bit-position 32(MSB). We have $V_{31}' = X_{31}' = Y_{31}' = Z_{31}' = R_{31}' = 0$, thus we require $V_{32}' \oplus X_{32}' \oplus Y_{32}' \oplus Z_{32}' \oplus R_{32}' = 0'$ which is not the case. Thus we have shown that even by using the additional degrees of freedom in the message expansion(*i.e.* the carry effect), we can never arrive at the desired difference `1b022000` in W_{17}. □

B L-characteristic for a 31-step collision of SHA-256-3R

The L-characteristic for 31-step SHA-256-3R including the message difference used in Sect. 3.2 is given in Table 5.

C Example of an 18-step collision for SHA-256

In Table 7 we give an example of an 18-step collision for SHA-256. We used a combination of the message modification technique described in Sect. 4 and the search for neutral bits to

- find the first 18-step collision in much less than a minute
- generate millions of them by using a large set of 2-neutral bits

The L-characteristic for this 18-step collision in SHA-256 including the message difference is given in Table 6. Since there are no conditions on the IVs, every IV including the standard-IV can be used. By adding more steps to this characteristic, near-collisions for more than 18-step can be derived in a straightforward manner. Note however that the weight of the difference at the output will be higher than one, thus a 1-near-collision as presented in Sect. 3.5 cannot be derived that way.

Table 5. L-characteristic for a 31-step collision in SHA-256-3R

Step	W’	A’	B’	C’	D’	E’	F’	G’	H’
01	00000001	00000001	0	0	0	00000001	0	0	0
02	66284480	22004000	00000001	0	0	62084400	00000001	0	0
03	8c2760a2	00000001	22004000	00000001	0	0981008b	62084400	00000001	0
04	95c7e0f6	28089550	00000001	22004000	00000001	68009150	0981008b	62084400	00000001
05	e9732fd2	0040a000	28089550	00000001	22004000	829685b1	68009150	0981008b	62084400
06	5be0be03	0	0040a000	28089550	00000001	20906a04	829685b1	68009150	0981008b
07	11b2513e	0	0	0040a000	28089550	00000001	20906a04	829685b1	68009150
08	6c2091d0	0	0	0	0040a000	28089550	00000001	20906a04	829685b1
09	4e794ee2	22004000	0	0	0	2240e000	28089550	00000001	20906a04
10	09ec2102	00000001	22004000	0	0	0981008b	2240e000	28089550	00000001
11	bdcf75a7	0	00000001	22004000	0	40080400	0981008b	2240e000	28089550
12	ad02b460	0	0	00000001	22004000	0	40080400	0981008b	2240e000
13	2240e000	0	0	0	00000001	22004000	0	40080400	0981008b
14	092d4192	0	0	0	0	00000001	22004000	0	40080400
15	44280480	0	0	0	0	0	00000001	22004000	0
16	0	0	0	0	0	0	0	00000001	22004000
17	36044000	14040000	0	0	0	14040000	0	0	00000001
18	175330fb	0	14040000	0	0	1501a070	14040000	0	0
19	4e869ebe	00000001	0	14040000	0	00000001	1501a070	14040000	0
20	44280480	0	00000001	0	14040000	40080400	00000001	1501a070	14040000
21	910e2130	0	0	00000001	0	14040000	40080400	00000001	1501a070
22	175330fa	0	0	0	00000001	0	14040000	40080400	00000001
23	00000001	0	0	0	0	00000001	0	14040000	40080400
24	44280480	0	0	0	0	0	00000001	0	14040000
25	14040000	0	0	0	0	0	0	00000001	0
26	0	0	0	0	0	0	0	0	00000001
27	00000001	0	0	0	0	0	0	0	0
28-31	0	0	0	0	0	0	0	0	0

Table 6. L-characteristic of an 18-step collision in SHA-256

Step	W’	A’	B’	C’	D’	E’	F’	G’	H’
01-03	0	0	0	0	0	0	0	0	0
04	80000000	80000000	0	0	0	80000000	0	0	0
05	22140240	0	80000000	0	0	20040200	80000000	0	0
06	42851098	0	0	80000000	0	0	20040200	80000000	0
07	0	0	0	0	80000000	0	0	20040200	80000000
08	80000000	0	0	0	0	80000000	0	0	20040200
09	22140240	0	0	0	0	0	80000000	0	0
10	0	0	0	0	0	0	0	80000000	0
11	0	0	0	0	0	0	0	0	80000000
12	80000000	0	0	0	0	0	0	0	0
13-18	0	0	0	0	0	0	0	0	0

Table 7. Example of an 18-step collision using the standard IV

i	M_i							
1-8	02679857	0183b9a1	005de4f5	0266ee0c	0d1442f0	06373a71	c445dec2	12542ec1
9-16	0982b61a	205a614c	2495a094	166ae4ac	15917909	1178f05a	0aae5a46	178058c6

D Pseudo-collisions for the compression function of SHA-256

We give an example of a pseudo-collision for the compression function of step-reduced SHA-256. The attacker has more freedom in such a setting: In addition to choose different messages M and M^*, he is also allowed to choose different IVs for the compression function. The goal is to find (M, M^*, IV, IV^*) such that $compress(M, IV) = compress(M^*, IV^*)$. Note that this type of pseudo-collision is different from the one described in [21].

The difference to Sect. 3.3 is that we have more degrees of freedom since we do not require the starting difference to be all-zero. To derive the actual collision, we used the same techniques as in Appendix C.

Note that this serves as an example. More steps can be achieved by extending the given characteristic in the backwards direction. In the example of a pseudo-collision given in Table 9, we need a different IV. $IV_{new} = IV_{standard} \oplus IV_{Corr}$. IV_{Corr} is given in Table 10. The corresponding L-characteristic is given in Table 8. The required difference in the IV for this pseudo-collision is given in this table as well. Note that this L-characteristic is similar to the 23-step related-key characteristic used in [9].

Table 8. L-characteristic for a 22-step pseudo-collision in SHA-256

Step	W'	A'	B'	C'	D'	E'	F'	G'	H'
IV'	0	00000200	0	0	50090088	0	0	20880000	10080080
01	0	0	00000200	0	0	40010008	0	0	20880000
02	0	0	0	00000200	0	0	40010008	0	0
03	0	0	0	0	00000200	0	0	40010008	0
04	0	0	0	0	0	00000200	0	0	40010008
05	0	0	0	0	0	0	00000200	0	0
06	0	0	0	0	0	0	0	00000200	0
07	0	0	0	0	0	0	0	0	00000200
08	00000200	0	0	0	0	0	0	0	0
09-22	0	0	0	0	0	0	0	0	0

Table 9. 22-step Pseudo-Collision with $M' \neq 0$

i	M_i							
1-8	39b1309b	048a8b67	02e0fc89	1dd4b937	02784cbd	1527473f	0134eb90	023f18aa
9-16	008a6849	063fbdbc	2e06da49	0f2e9e2a	085d407e	1686fa83	03ad81fe	091da09b

Table 10. IV Correction for 22-step Pseudo-Collision with $M' \neq 0$

i	IV_{Corr}				
1-5	DCBD1A68	00000000	00000000	00000080	60810000